
IBM Developer
SKILLS NETWORK
Winning Space Race
with Data Science
AJALI AUGUSTINE E.
19th DECEMBER 2023
IBM_Data_science_cert|SpaceX

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

SUMMARY OF METHODOLOGY:

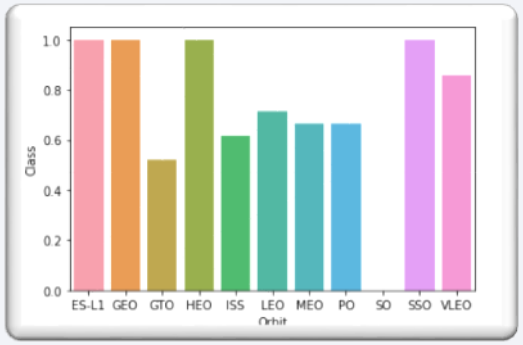
The project was done by following the steps below:

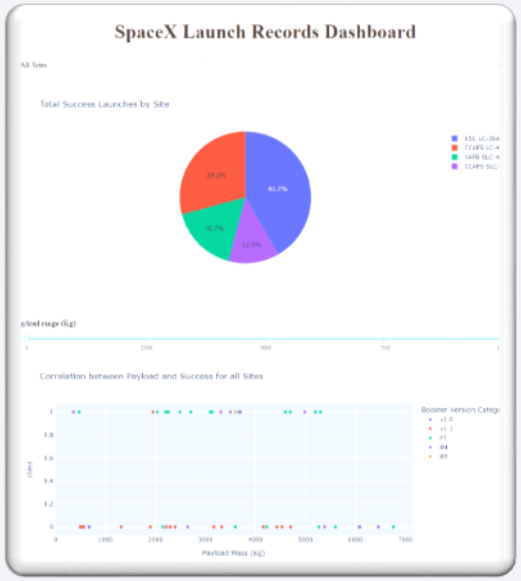
- Data Collection
- Data processing
- Exploratory data analysis (EDA) using SQL and python
- Interactive visualizations using Plotly and Follium
- Predictive Analysis (Building Classifiers)

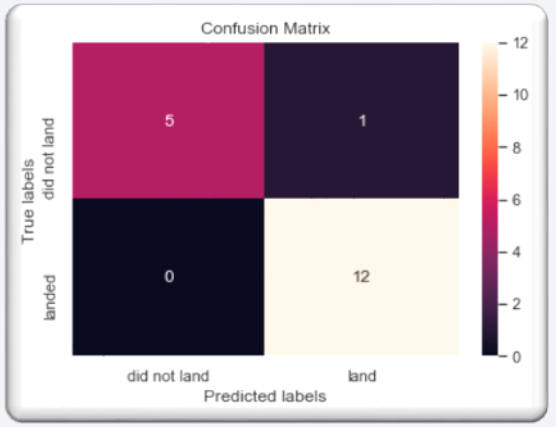
SUMMARY OF RESULTS:

The following output was gotten from this projects:

1. EDA results
2. Geospatial analytics results
3. Interactive dashboards
4. Predictive analysis of classifications models







Introduction

Revolutionizing Rocket Launch Economics with Machine Learning

- In an era where space exploration meets commercial viability, cost efficiency is a defining factor. SpaceX has pioneered a paradigm shift with its Falcon 9 rocket, offering launches at a fraction of the cost compared to competitors—$62 million versus $165 million. A key enabler of this cost advantage is SpaceX's groundbreaking reuse of the first rocket stage.
- This presentation introduces a cutting-edge machine learning pipeline designed to predict the success of Falcon 9 first-stage landings. The implications are profound for companies aiming to challenge SpaceX's dominance in the market. By harnessing predictive insights, competitors can strategically bid for rocket launches, estimating costs based on the likelihood of first-stage recovery.
- If we can make predictions to determine whether the first stage will land successfully, then we can evaluate the cost of a launch, and use this information to assess whether an alternate company should bid and SpaceX for a rocket launch.
- Join us as we delve into the methodology, benefits, and potential market impact of this innovative approach to rocket launch economics.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Making GET requests to the SpaceX API
  - Using web-scraping to extract data
- Perform data wrangling
  - Using the .fillna() method to remove NAN values
  - Count distinct values
  - Drop some irrelevant columns
  - Added new columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Built some classifications models (KNN, Logical Regression, Decision Tress and)
  - Trian the models using the Trains datasets
  - Test the models and made predictions on the test datasets
  - Evaluated the accuracy of our models using BestScore and Score methods
  - Created a table from the accuracy score gotten from each classification's models
  - Created a bar chats to visualize the score and to pick the best model.

Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

The following steps was followed to retrieve SpaceX lunch data from SpaceX API

1
- Make a GET response to the SpaceX REST API
- Convert the response to a .json file then to a Pandas DataFrame

2
We use traditional methods to clean the data
- Created lists for data to be stored in
- Call necessary functions to retrieve and fill data into the lists created
- constructed the dataset using the list as values in the dictionary

3
- Create a Pandas DataFrame from the constructed dictionary dataset

4
- Filter the DataFrame to only include Falcon 9 launches
- Reset the FlightNumber column
- Replace missing values of PayloadMass with the mean PayloadMass value

- **Github url** :

1
```
static_json_url='https://cf-courses-data.s3.us.cloud
response = requests.get(static_json_url)
response.json()
```

2
```
# Use json_normalize meethod to convert the json result
data = pd.json_normalize(response.json())
```

3
```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
set of our dataframe keeping only the features
ket', 'payloads', 'launchpad', 'cores', 'flig

rows with multiple cores because those are fa
'cores'].map(len)==1]
'payloads'].map(len)==1]

and cores are lists of size 1 we will also ext
ata['cores'].map(lambda x : x[0])
= data['payloads'].map(lambda x : x[0])
```

```
# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.head()
```

4
```
data_falcon9.isnull().sum()
```

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean().round(1)
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, mean, inplace=True)
```

Data Collection – Web Scraping

Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

1
- Requested the HTML page from the static URL
- Assigned the response to an object

2
- Created a BeautifulSoup object from the HTML response object
- Find and scrapped all tables within the HTML page

3
- Collect all column header names from the tables found within the HTML page

4
- Use the column names as keys in a dictionary
- Use custom functions and logic to parse all launch tables to fill the dictionary values

5
- Convert the dictionary to a Pandas DataFrame ready for export

GitHub URL:

1
```
Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean().round(1)
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, mean, inplace=True)
```

2
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

3
```
umn_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for row in first_launch_table.find_all('th'):
  name = extract_column_from_header(row)
  if (name != None and len(name) > 0):
    column_names.append(name)
```

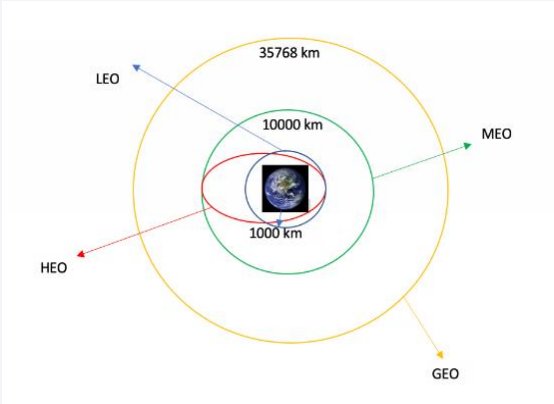
4
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty l
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

5
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.head()
```

Data Wrangling - Pandas

Context:

- The SpaceX dataset contains several Space X launch facilities, and each location is in the Launch Site column.
- Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the image below . The orbit type is in the Orbit column.



Initial Data Exploration:

utilizing the **.value_counts()** method we were able to determine the following:

- Number of launches on each site
- Number and occurrence of each orbit
- Number and of landing outcome per orbit type occurrence

1
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO       27
ISS       21
VLEO      14
PO         9
LEO        7
SSO        5
MEO        3
ES-L1      1
HEO        1
SO         1
GEO        1
```

3
```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
```

```
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
True Ocean       5
False Ocean      2
None ASDS        2
False RTLS       1
Name: Outcome, dtype: int64
```

Data Wrangling - Pandas

Context:

- The landing outcome of each launch is recorded in the Outcome column some classifications of the outcomes are listed and explained below:
  - True Ocean – the mission outcome was successfully landed on the expected region of the ocean
  - False Ocean – the mission outcome was unsuccessfully landed, thus landing on a different region of the ocean or didn't land at all.
  - True RTLS – the mission outcome was successfully landed on a ground pad as expected
  - False RTLS – the mission outcome was unsuccessfully landed on a ground pad.
  - True ASDS – the mission outcome was successfully landed on a drone ship as expected
  - False ASDS – the mission outcome was unsuccessfully landed on a drone ship.
  - None ASDS and None None – these categories represent a failure to land.

Data Wrangling:

- To determine whether a booster will successfully land, we have to create separate column with only binary values, i.e., where the value is 1 represents successful landing or 0 representing the unsuccessful landing of the first stage.
- This was achieved by:
  1. Defining a set of unsuccessful (bad) outcomes, bad_outcome
  2. Creating a list called landing_class, using a for loop where landing is unsuccessful in the corresponding row on Outcome column zero (0) is appended on Landing_class Column, otherwise, one (1) is appended.
  3. Create a column that contains the values from the list landing_class
  4. Export the DataFrame as a .csv file.

Github URL:

1

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

2

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
  if outcome in bad_outcomes:
    landing_class.append(0)
  else:
    landing_class.append(1)

print(landing_class)
```

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
```

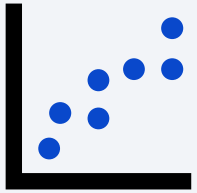
3

```
df['Class']=landing_class
df[['Class']].head(8)
```

EDA with Data Visualization

SCATTER PLOTS

- Scatter charts were produced to investigate the kind of correlations that exist between columns to determine if it linear or not, some of the columns are:

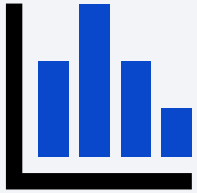


- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

Scatter charts are useful to observe relationships, or correlations, between two numeric variables.

BAR PLOTS

- A bar chart was created to visualize the relationship between the two columns below:
- Success Rate and Orbit Type



Bar charts are used to compare a numerical value to a categorical variable. Horizontal or vertical bar charts can be used, depending on the size of the data.

LINE PLOTS

- Line charts were produced to visualize the trend between the columns below over time:
- Success Rate and Year (i.e. the launch success yearly trend)

Line charts contain numerical values on both axis and are generally used to show the change of a variable over time.

EDA with SQL

To further explore the datasets, SQL statements were run retrieve the following information.

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string ‘CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Build an Interactive Map with Folium

The following steps were taken to visualize the launch data on an interactive map:

1. Mark all launch sites on a map
   - Initialise the map using a Folium Map object
   - Add a folium.Circle and folium.Marker **for each launch site on the launch map**

2. Mark the success/failed launches for each site on a map
   - Since we have some launches with the same coordinates, it will be best to group them together.
   - Before we group them, we assigned a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
   - To put the launches into clusters, for each launch, add a folium.Marker to the MarkerCluster() object.
   - Create an icon as a text label, assigning the icon_color as the marker_colour determined previously.

3. Calculate the distances between a launch site to its proximities
   - To explore the proximities of launch sites, calculations of distances between points can be made using the Lat and Long values.
   - After marking a point using the Lat and Long values, create a folium.Marker object to show the distance.
   - To display the distance line between two points, draw a folium.PolyLine and add this to the map.

Build a Dashboard with Plotly Dash

- The following plots were added to a Plotly Dash dashboard to have an interactive visualisation of the data:

1. Added Pie chart (px.pie()) to show the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a dcc.Dropdown() object) to see the success/failure ratio for an individual site

2. Scatter graph (px.scatter()) to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered (using a RangeSlider() object) by ranges of payload masses
   - It could also be filtered by booster version

Predictive Analysis (Classification)

The steps below were taking to build the classifiers, evaluate it, and find the best performing model :

Model Development

Model Evaluation

Finding the Best Classification Model

- **To prepare the dataset for model development we did the following**:
  - Loaded dataset
  - Performed necessary data transformations (standardise and pre-process)
  - Split dataset into training and test data sets, using train_test_split()
  - chose which of the machine learning algorithms are most appropriate
- **For the chosen algorithms we perform the following**:
  - Create a GridSearchCV object and a dictionary parameters
  - Fit the object to the parameters
  - Use the training data set to train the model

- **For chosen algorithms we perform the following:**
  - Used the output GridSearchCV object to:
    1. Check the tuned hyperparameters (best_params_)
    2. Check the accuracy (score and best_score_)
  - Plot and examine the Confusion Matrix

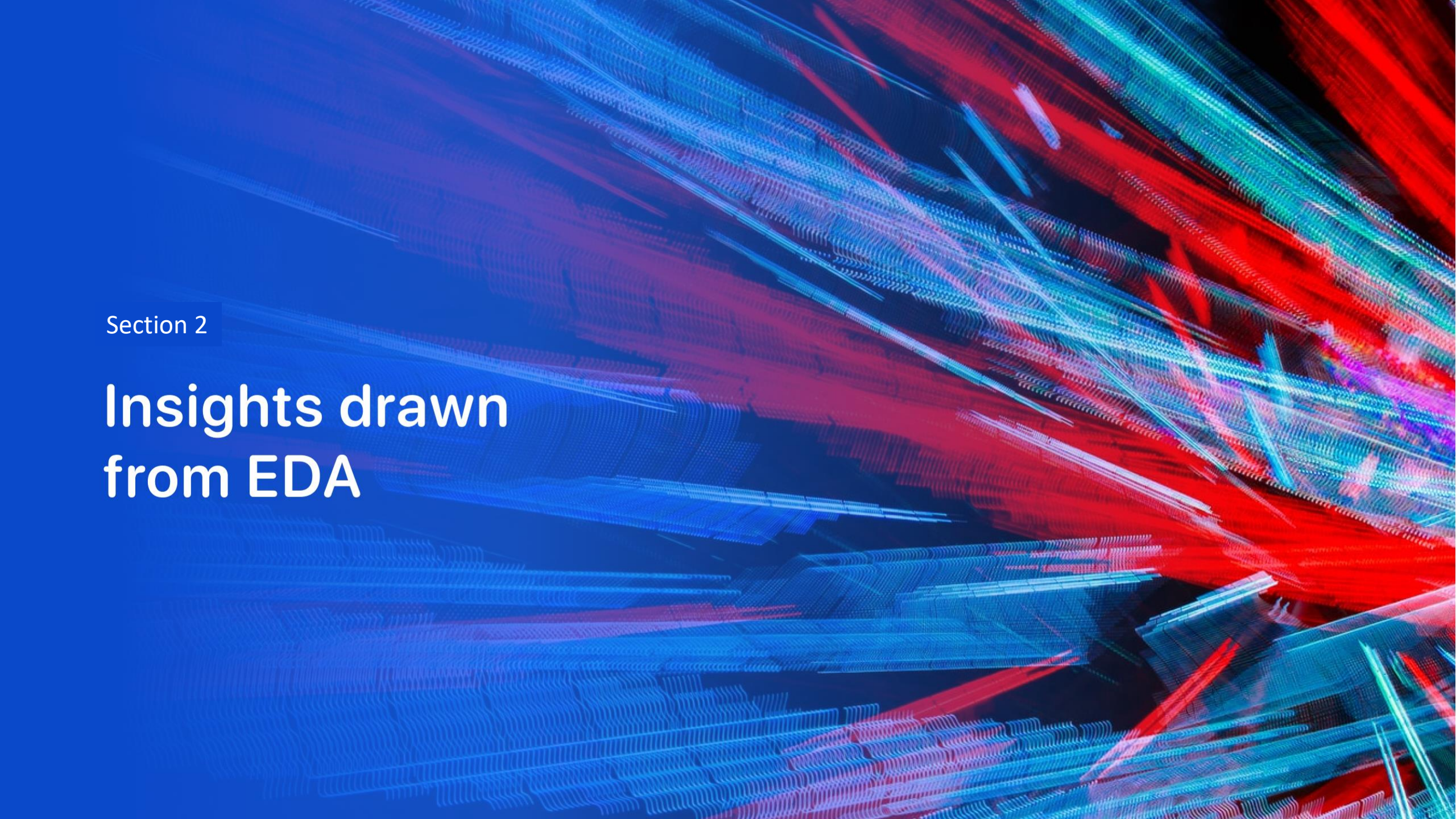
- Finally, the accuracy scores for all chosen algorithms was reviewed.
- The model with the highest accuracy score was chosen as the best performing model

Results

Exploratory Data Analysis

Interactive Analytics

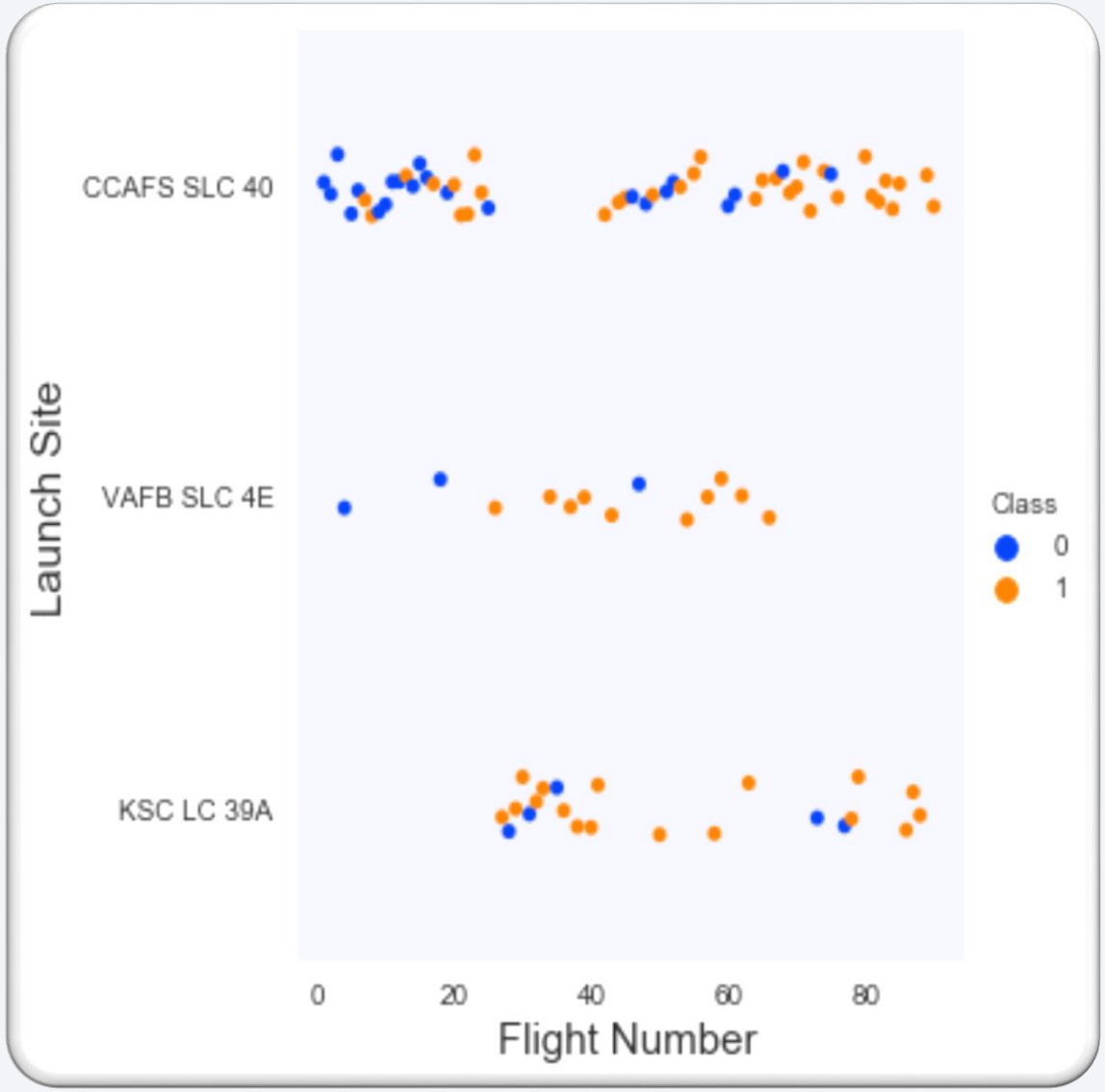
Predictive Analysis


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

The scatter plot of Launch Site vs. Flight Number shows that:

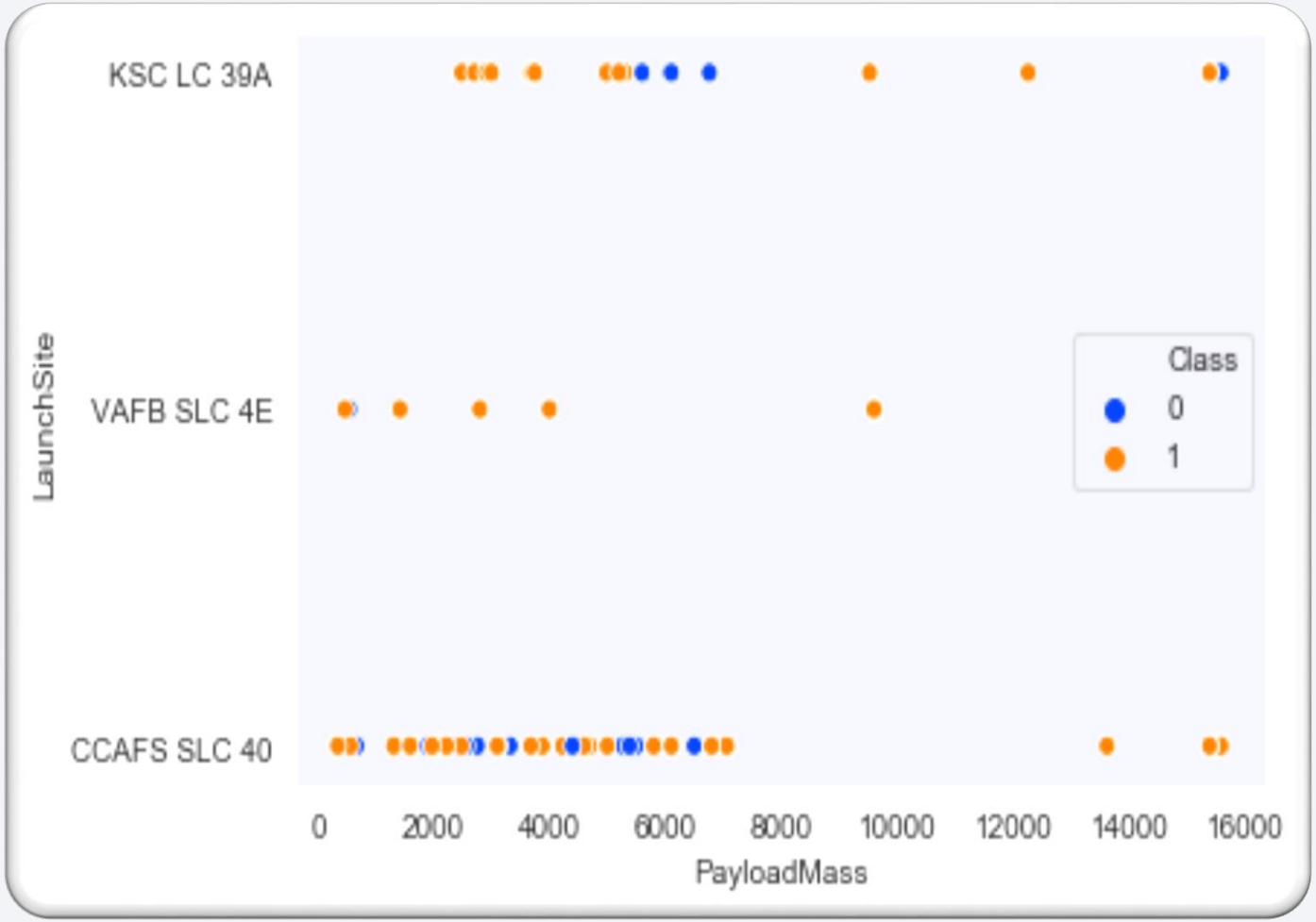
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40 and recorded much more unsuccessful launch outcomes.
- The flights from VAFB SLC 4E also show this trend, that earlier flights recorded less successful outcomes.
- There were no early flights launched from KSC LC 39A, so the launches from this site are more recent and more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).



Payload vs. Launch Site

From the scatter plot of Launch Site vs. Payload Mass we can deduce the following:

- For payload mass close to or greater than 7000 kg, there are very high number successful landings although we have very small data for these heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 and the fewest from VAFB SLC 4E.
- Generally, we can say that the rate of successful landing is higher for all landing sites.

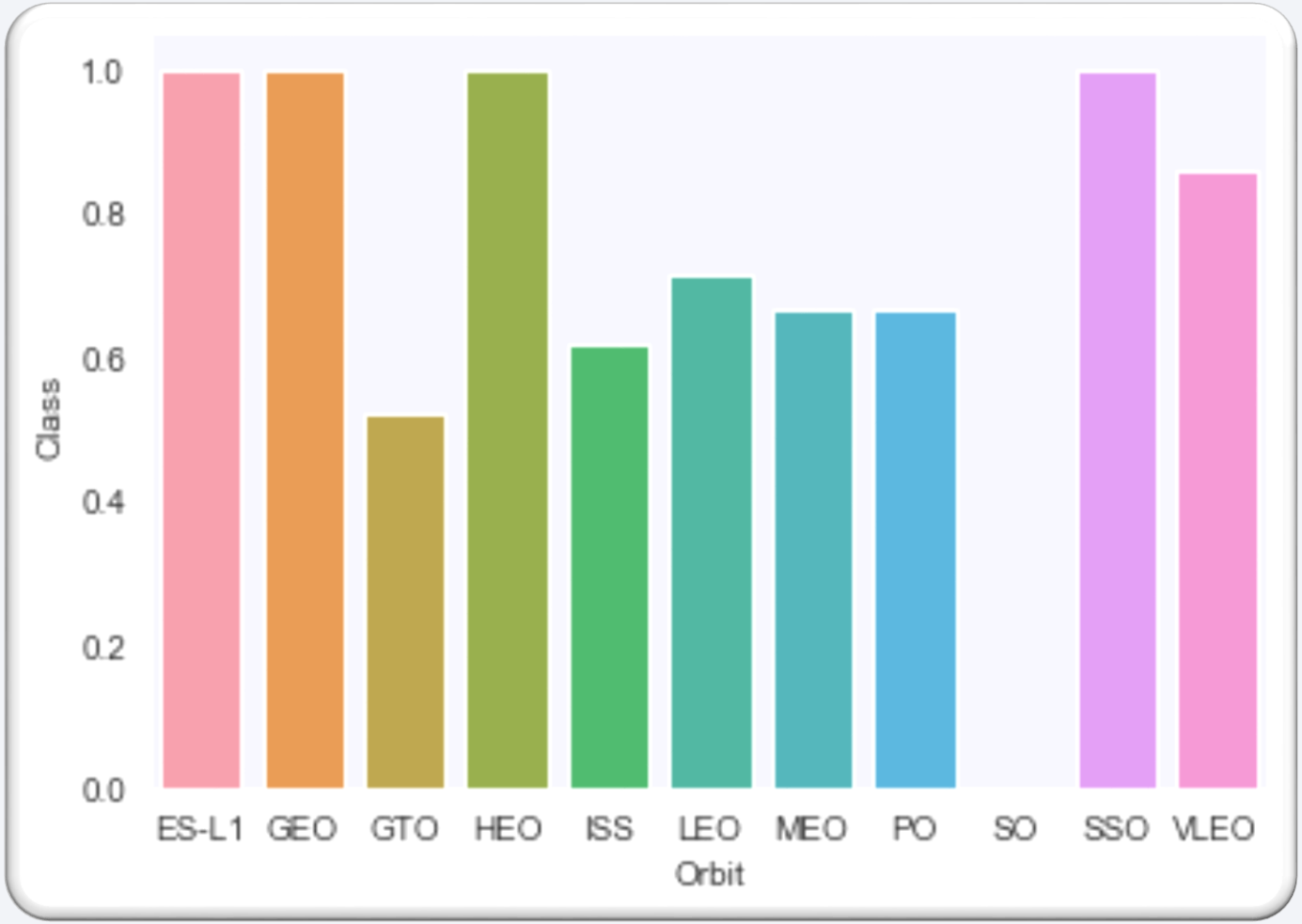


Success Rate vs. Orbit Type

From the bar chart of Success Rate vs. Orbit Type can deduce that the orbits listed below have the highest (100%) success rate:

- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

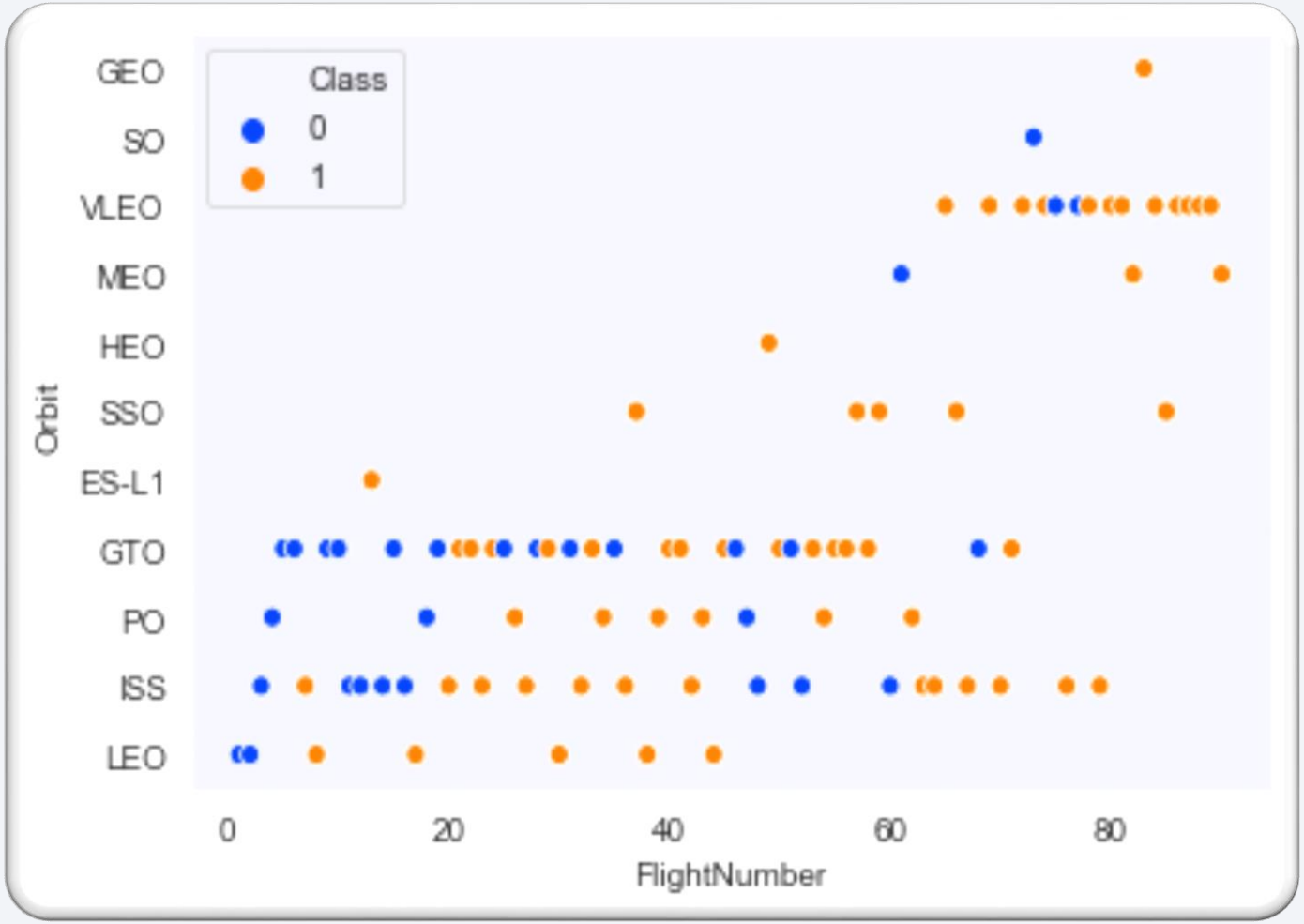
- The orbit with the lowest (0%) success rate is:
- SO (Heliocentric Orbit)
- However, a good number of orbit have a success rate greater than 50%



Flight Number vs. Orbit Type

This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:

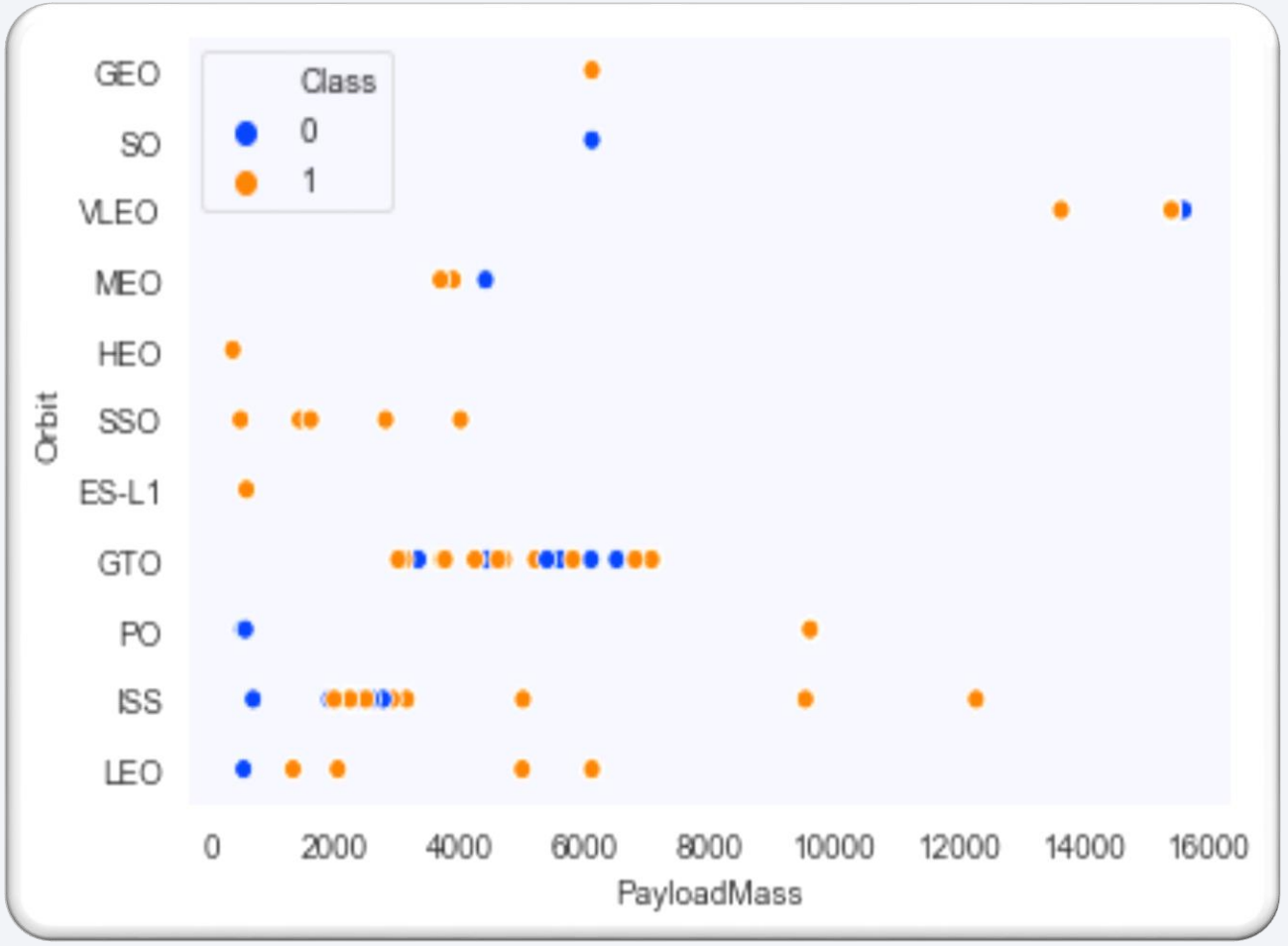
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).



Payload vs. Orbit Type

This scatter plot of Orbit Type vs. Payload Mass shows that:

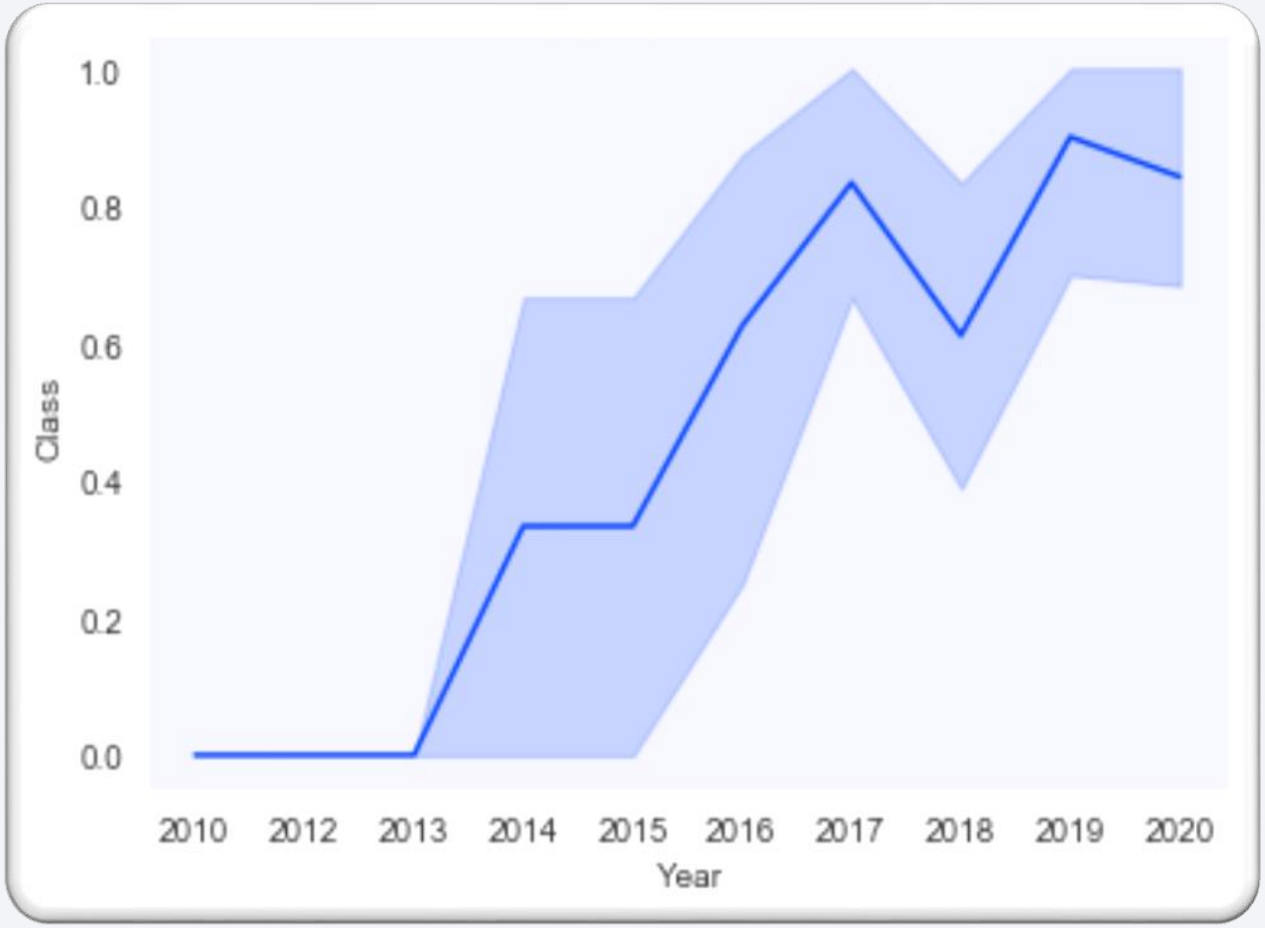
- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.



Launch Success Yearly Trend

The line chart of yearly average success rate shows that:

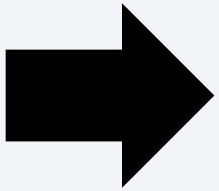
- All landings for the year 2010 to 2013 were generally unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased from 0% to about 37%, after which for the year 2014 to 2015 the success rate remained constant at 37%.
- From 2015 to 2017, Landing success rate progressively increased to about 80%, although during period of 2017 to 2018, landing success rate reduced to about 60% but increased again to 90% in 2019.
- Generally, after 2016, there was always a minimum of 60% chance of landing a rocket successfully.



All Launch Site Names

The SQl statements below was used to quarry the names of all unique launch sites.

```
%sql SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXTBL;
```



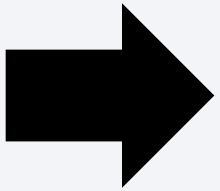
launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

The word **UNIQUE** returns only unique values from the **LAUNCH_SITE** column of the **SPACEXTBL** table.

Launch Site Names Begin with 'CCA'

The quarry below was executed to find 5 records where launch sites begin with ‘CCA’.

```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```



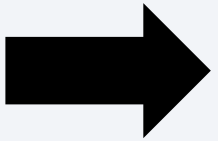
launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

LIMIT 5 Extracts only 5 records, and the **LIKE** keyword is used to specify the **‘CCA%’**, in this case we are to retrieve string values beginning with ‘CCA’.

Total Payload Mass

Calculate the total payload carried by boosters from NASA.

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```



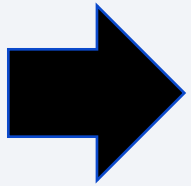
total_payload_mass
45596

The **SUM** keyword is used to calculate the total of the **LAUNCH** column, and the **SUM** keyword (and the associated condition) filters the results to only boosters from NASA (CRS).

Average Payload Mass by F9 v1.1

The quarry below was executed to calculate the average payload mass carried by booster version F9 v1.1.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```



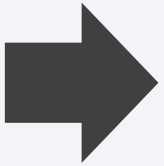
average_payload_mass
2928

The **AVG** keyword is used to calculate the average of the **PAYLOAD_MASS__KG_** column, and the **WHERE** keyword with its associated condition filters the results to only the F9 v1.1 booster version.

First Successful Ground Landing Date

Find the dates of the first successful ground landing outcomes.

```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL \
    WHERE LANDING__OUTCOME = 'Success (ground pad)';
```



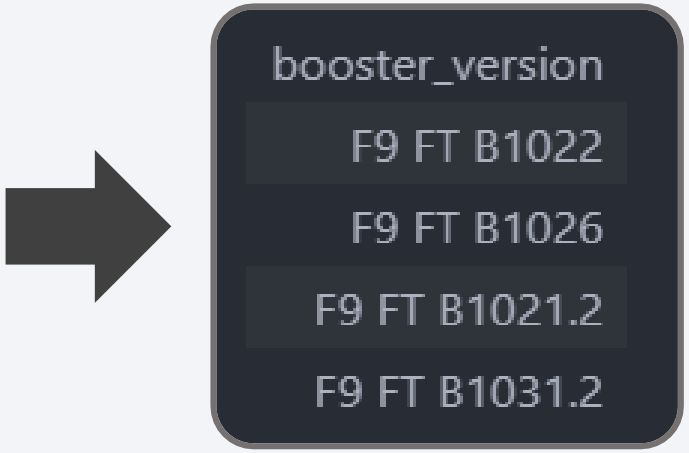
first_successful_ground_landing
2015-12-22

The MIN keyword is used to extract the minimum values of the DATE column, i.e. the earliest date, and the WHERE keyword and its associated conditions filters the results to only the successful ground landings.

Successful Drone Ship Landing with Payload between 4000 and 6000

The task was to List the names of successfully landed Boosters on drone ship, specifically targeting those with payload mass greater than 4000 but less than 6000.

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000);
```

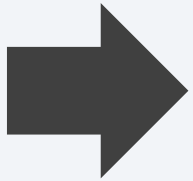


The **WHERE** keyword is used to filter the results to include only those that satisfy both conditions in the brackets, **AND** keyword was used because there are two distinct conditions. The **BETWEEN** keyword allows us to quarry values from 4000 to 6000.

Total Number of Successful and Failure Mission Outcomes

This task was to Calculate the total number of failed and successful mission outcome.

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```



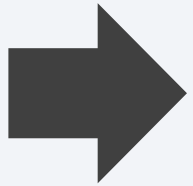
mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

The **COUNT** keyword was used to calculate the total number of mission outcomes, and the **GROUPBY** keyword is also used to group these results by the type of mission outcome.

Boosters Carried Maximum Payload

This task was to List the names of boosters which carries the maximum payload mass.

```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
    WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```



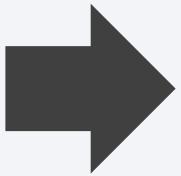
booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

A subquery is used here. The **SELECT** statement within the brackets finds the maximum payload, and this value is used in the **WHERE** condition. The **DISTINCT** keyword is then used to retrieve only distinct/unique booster versions.

2015 Launch Records

This task is to List the failed landing outcomes in drone ship, their booster versions and launch site names for only the year 2015.

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL \
    WHERE (LANDING__OUTCOME = 'Failure (drone ship)') AND (EXTRACT(YEAR FROM DATE) = '2015');
```



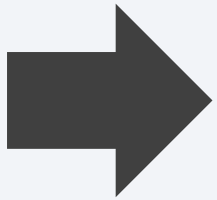
booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

The **WHERE** keyword is used to filter the results for only failed landing outcomes, **AND** only for the year of 2015.

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

This task is to rank the count of landing outcomes from 2010-06-04 to 2017-03-20 in descending order.

```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY LANDING__OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```



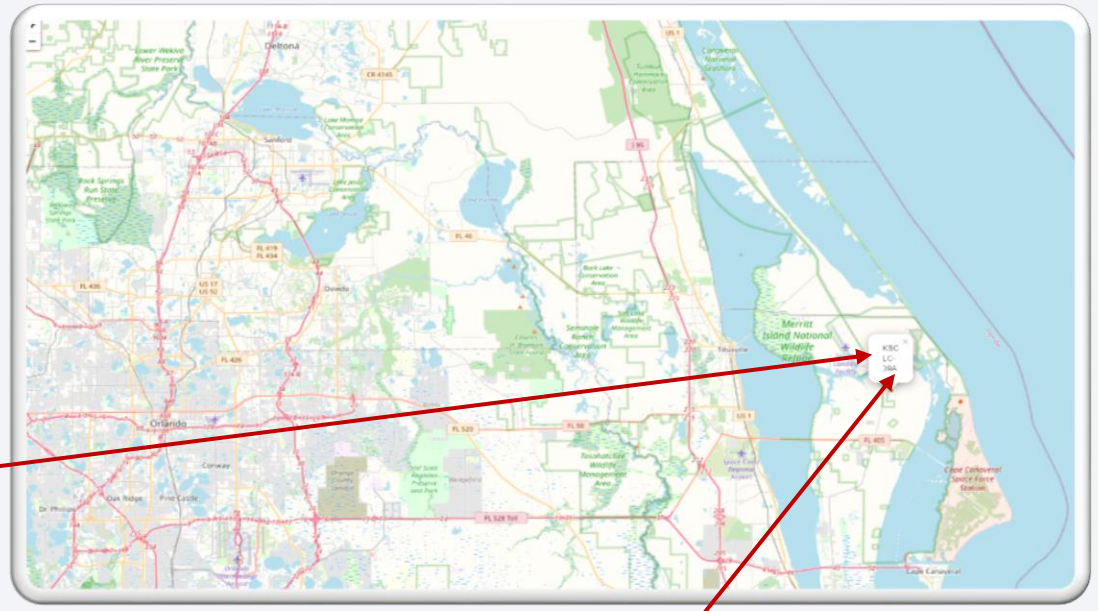
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

The **WHERE** keyword is used with the **BETWEEN** keyword to filter out the results to the dates specified. The results are then grouped and ordered, using the keywords **GROUP BY** and **ORDER BY**, respectively, where **DESC** is used to specify the descending order.

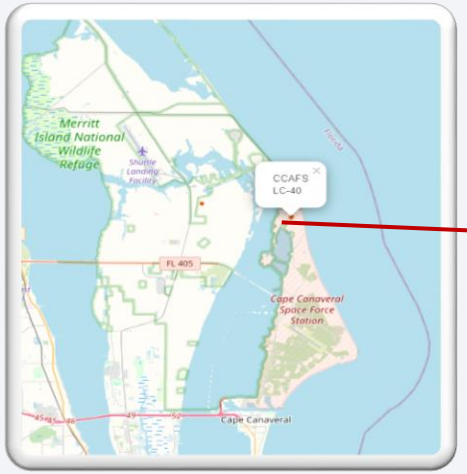
Section 3

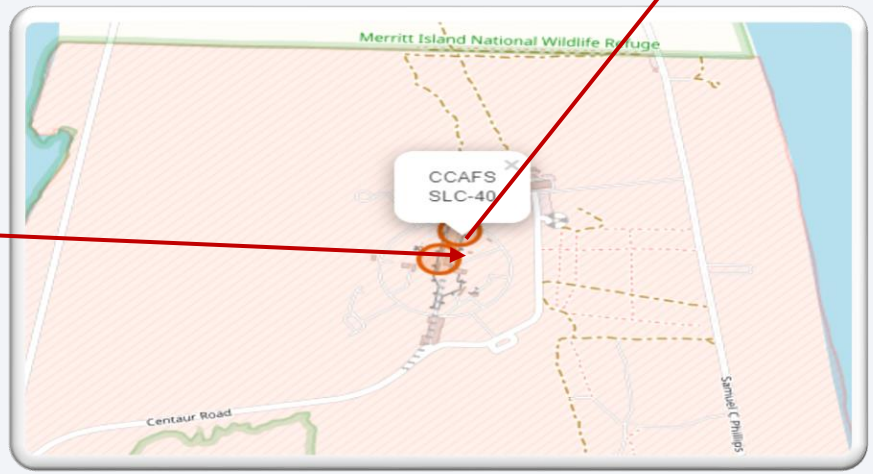
Launch Sites Proximities Analysis

All Launch Site on Folium Map

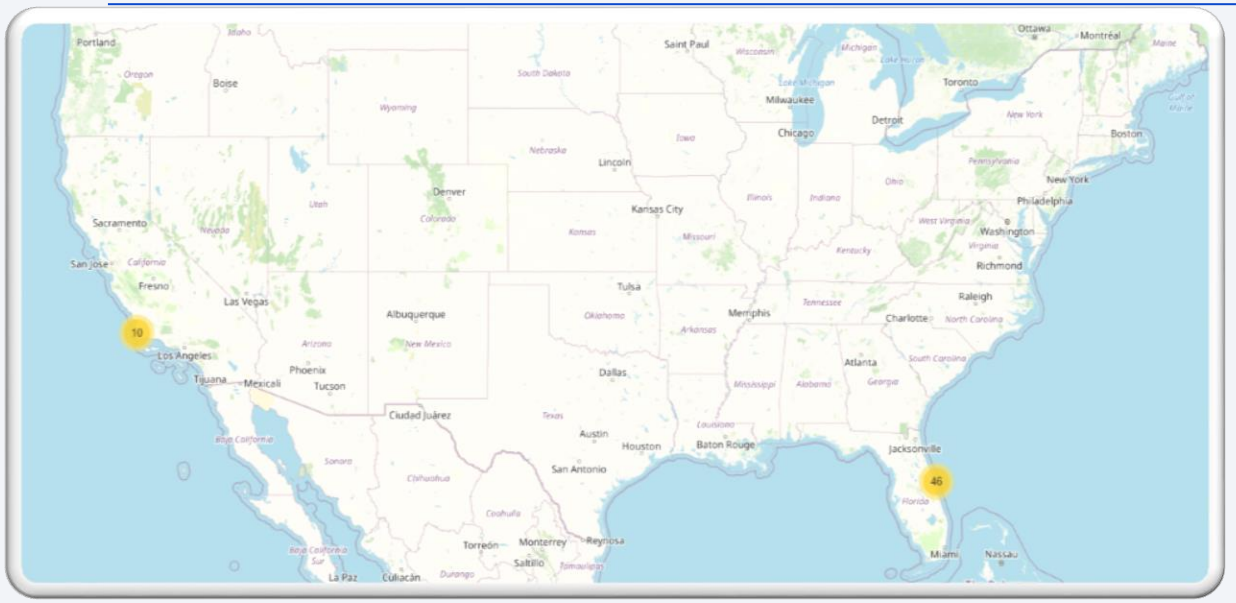


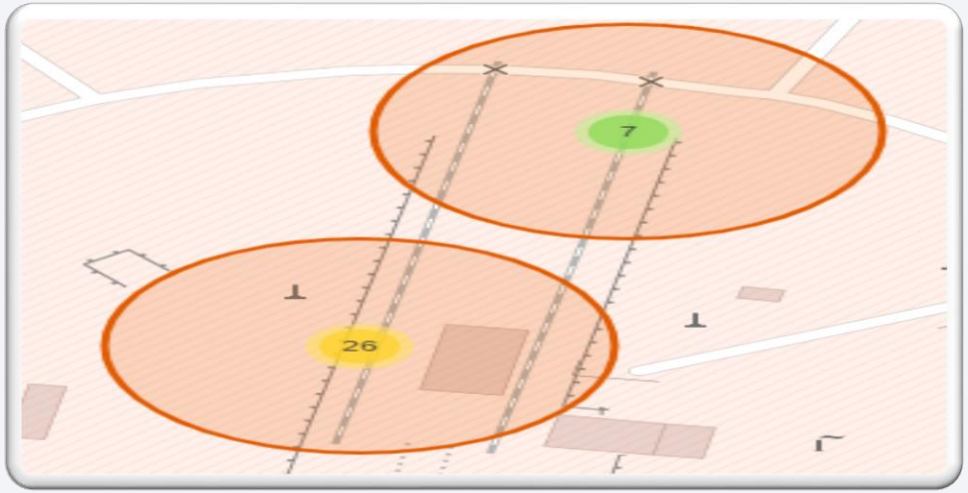
All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.



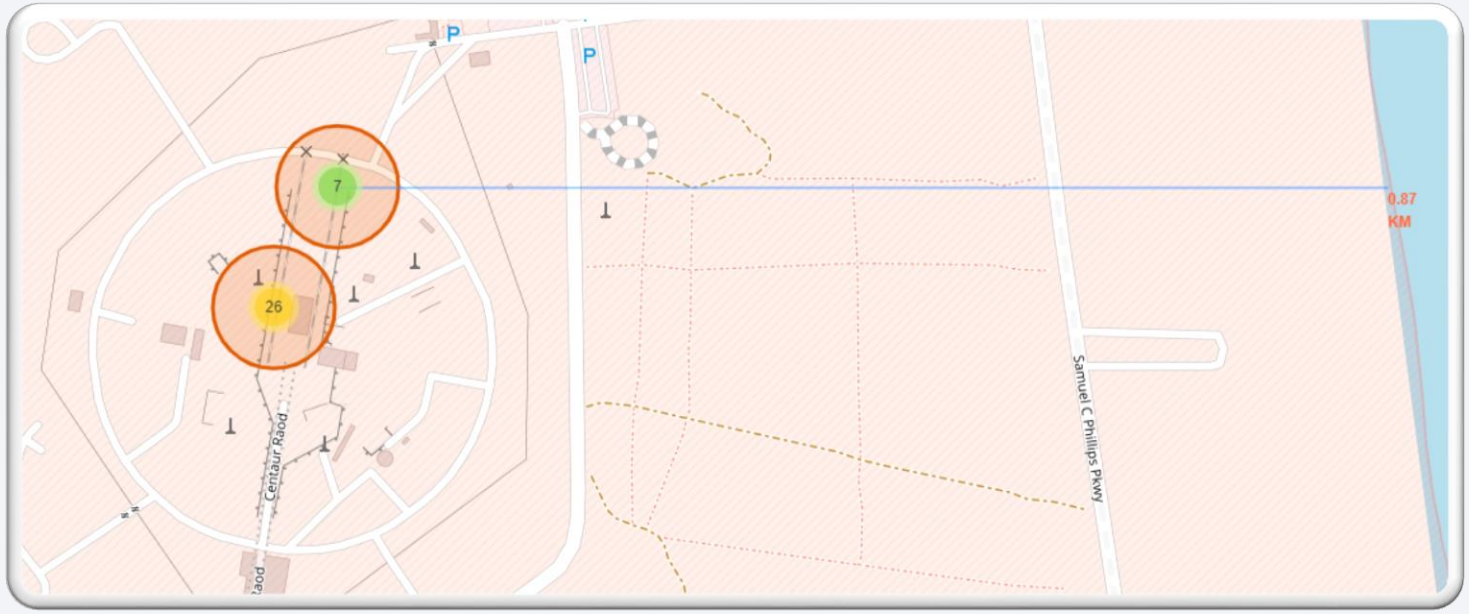


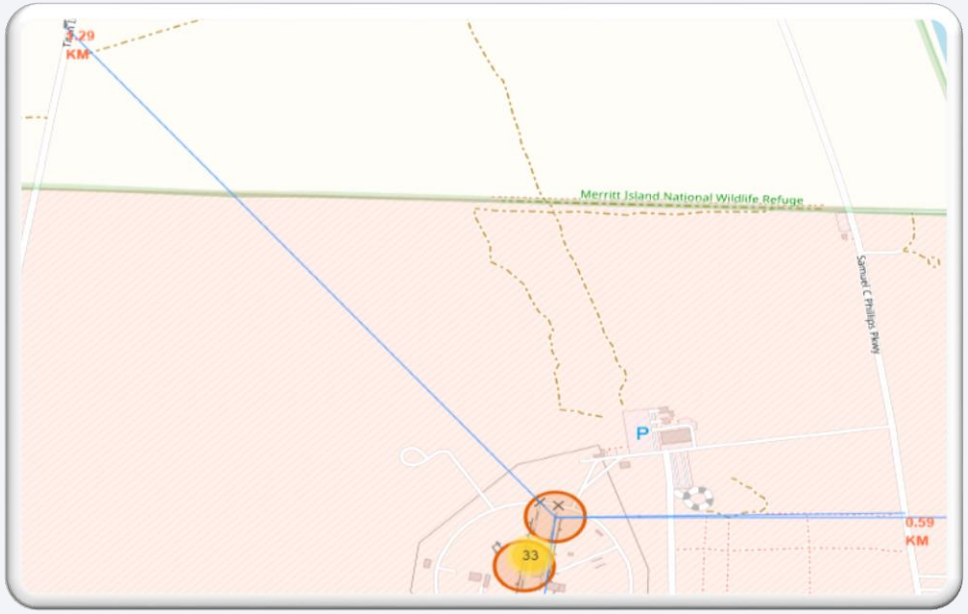
Map showing launch sites

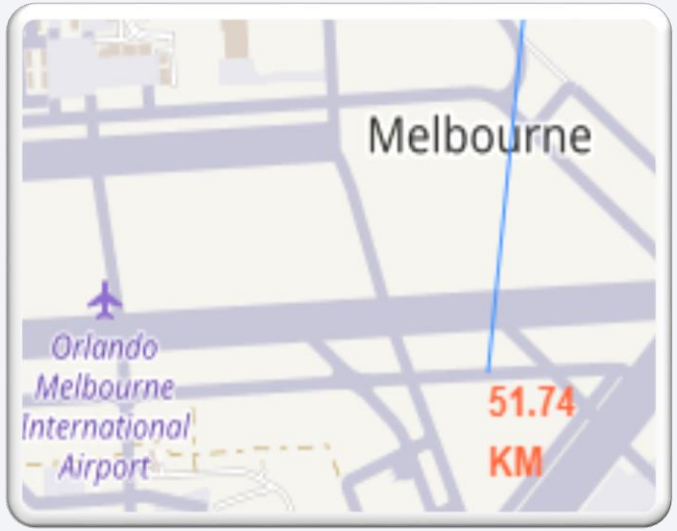




<Folium Map Screenshot 3>

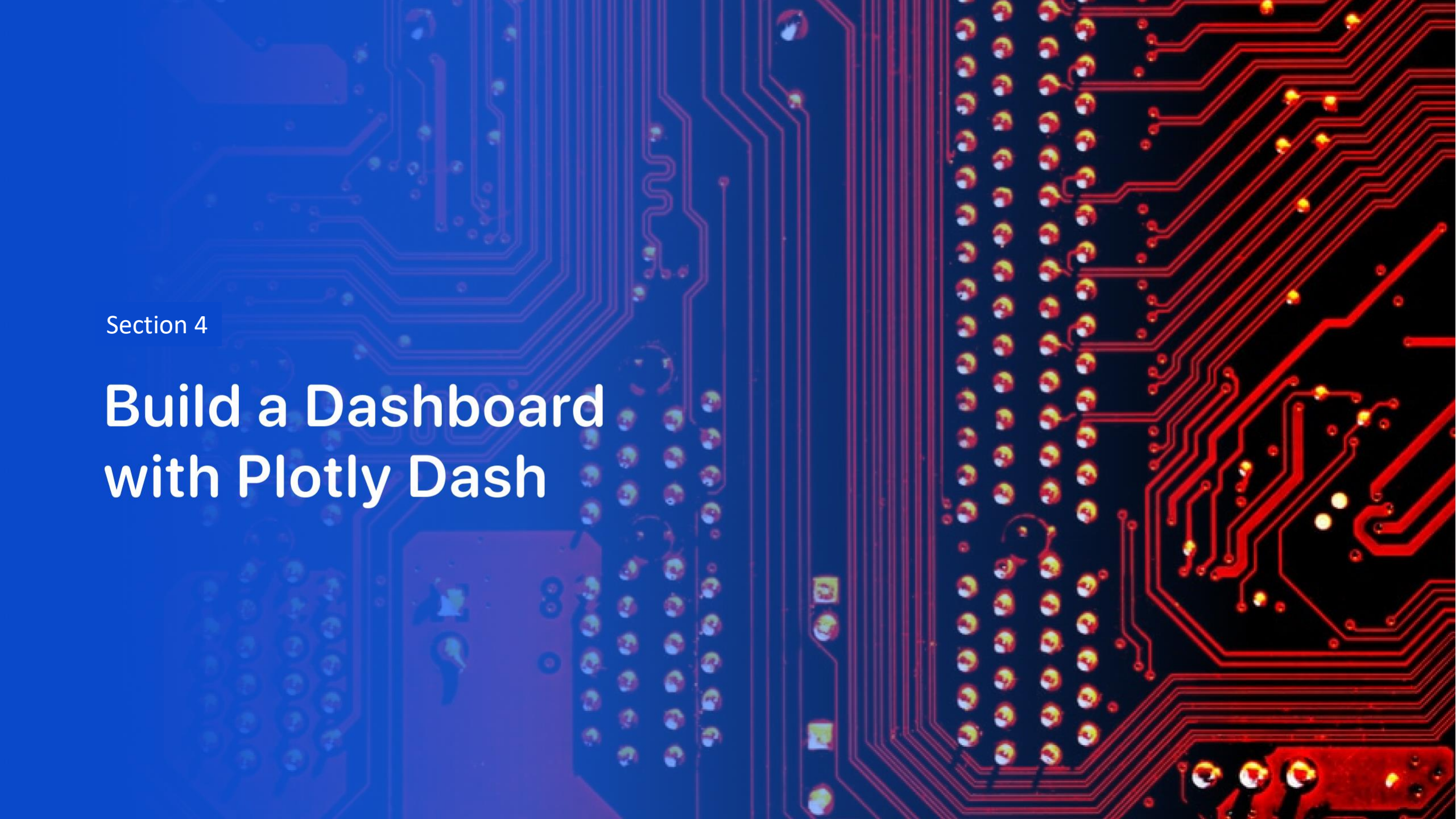




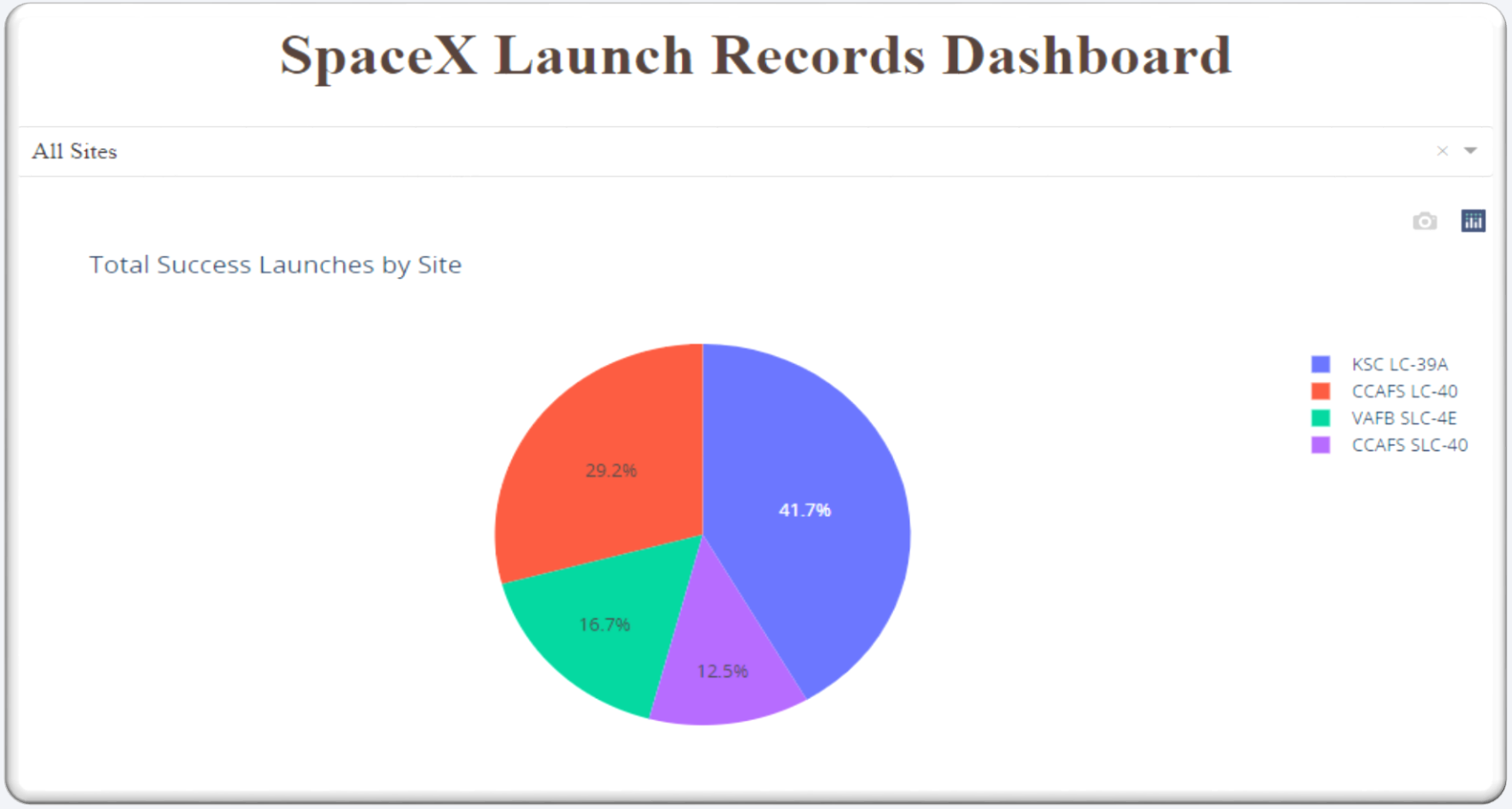


Using the CCAFS SLC-40 launch site as an example site, we can understand more about the placement of launch sites.

1. Are there launch sites close to railways? YES. The coastline is only 0.87 km due East.
2. Are there launch sites close to highways? YES. The nearest highway is only 0.59km away.
3. Are there launch sites close to railways? YES. The nearest railway is only 1.29 km away.
4. Do launch sites keep reasonable distance away from cities? YES. The nearest city is 51.74 km away.

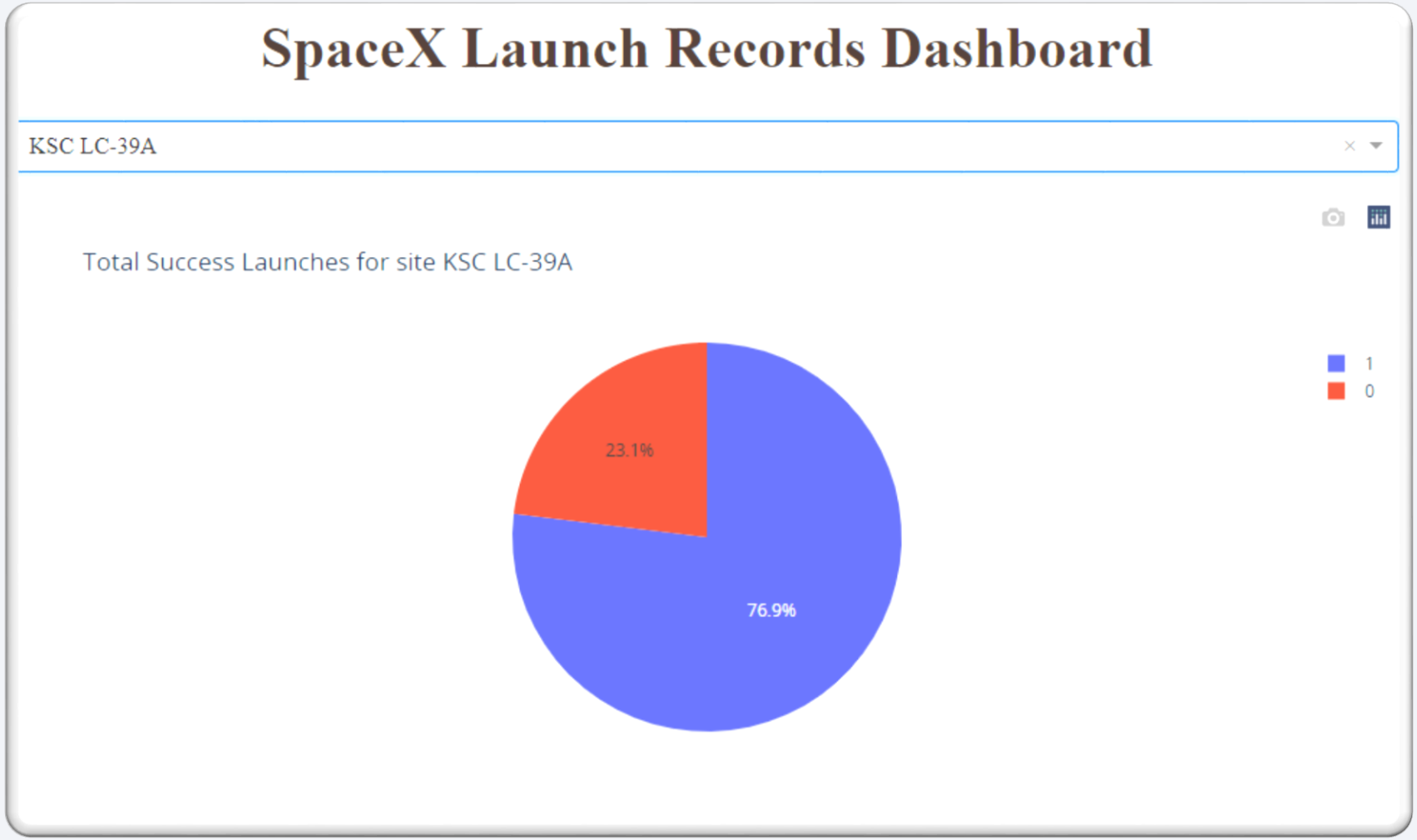

Section 4
Build a Dashboard with Plotly Dash

Spacex launch Success by Site



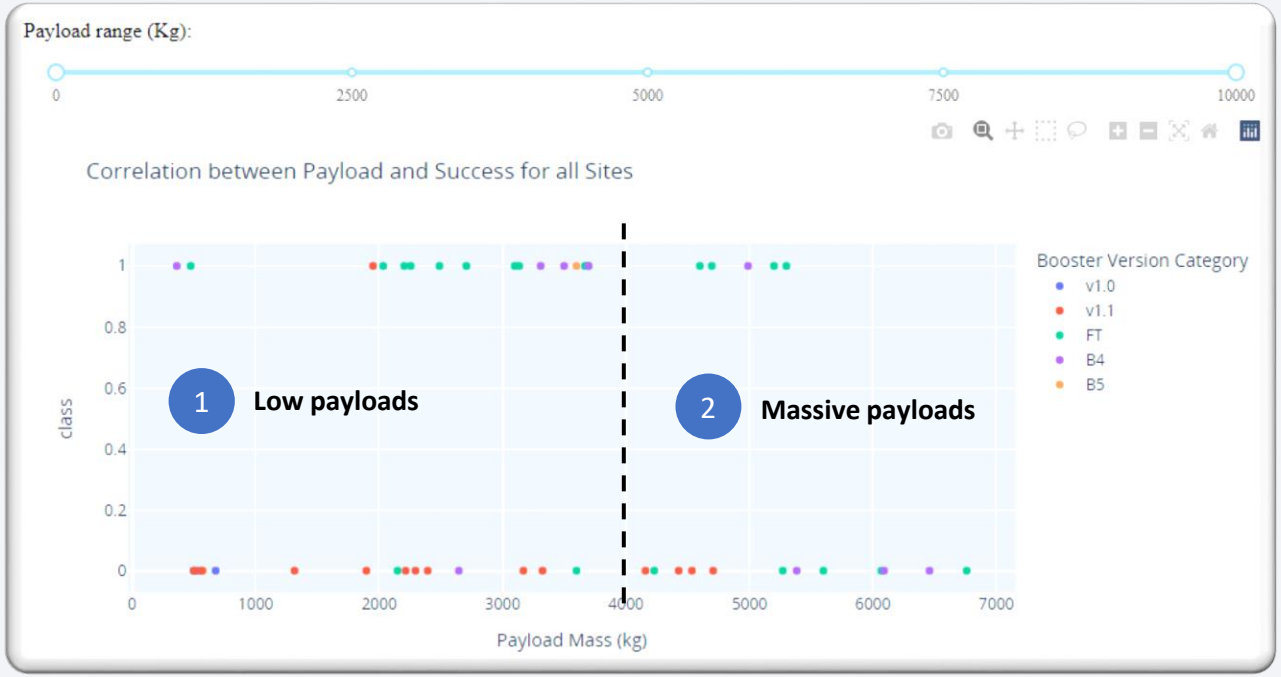
The launch site KSC LC-39 A recorded the most successful launches, with 41.7% of the total successful launch and CCAPS LC-40 was next with a recorded success rate of about 29.2%.

<Dashboard Screenshot 2>



KSC LC-39 A happened to be the launch site with the highest success rate of about 76.9% and a failure rate of about 23.1%

Launch Outcome VS. Payload scatter plot for all sites



Plotting the launch outcome vs. payload for all sites shows a gap around 4000 kg, so it makes sense to split the data into 2 ranges:

- **0 – 4000 kg (low payloads)**
- **4000 – 10000 kg (massive payloads)**

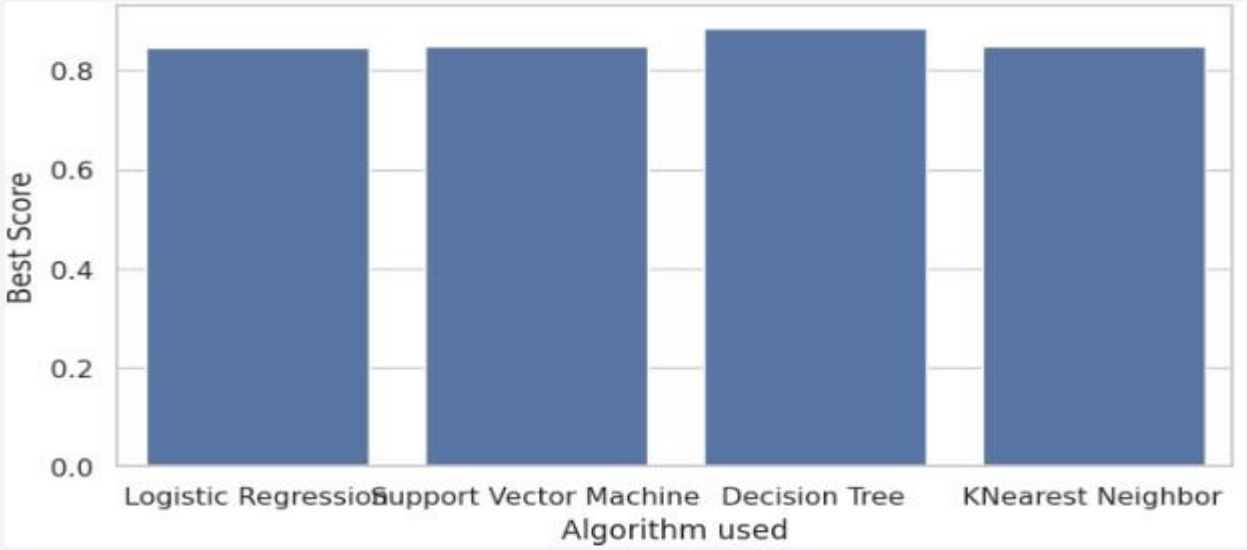
After splitting we can see that the success for massive payloads is lower than that for low payloads.

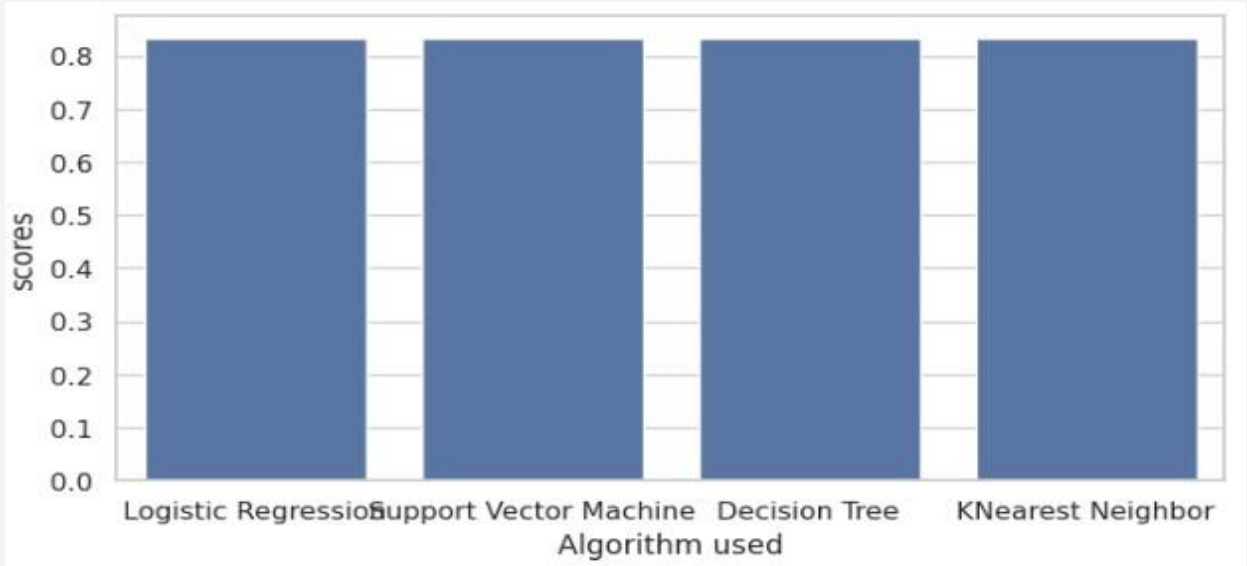
It should also be noted that some booster types (v1.0 and B5) have not been launched with massive payloads yet.

Section 5

Predictive Analysis (Classification)

Classification Accuracy





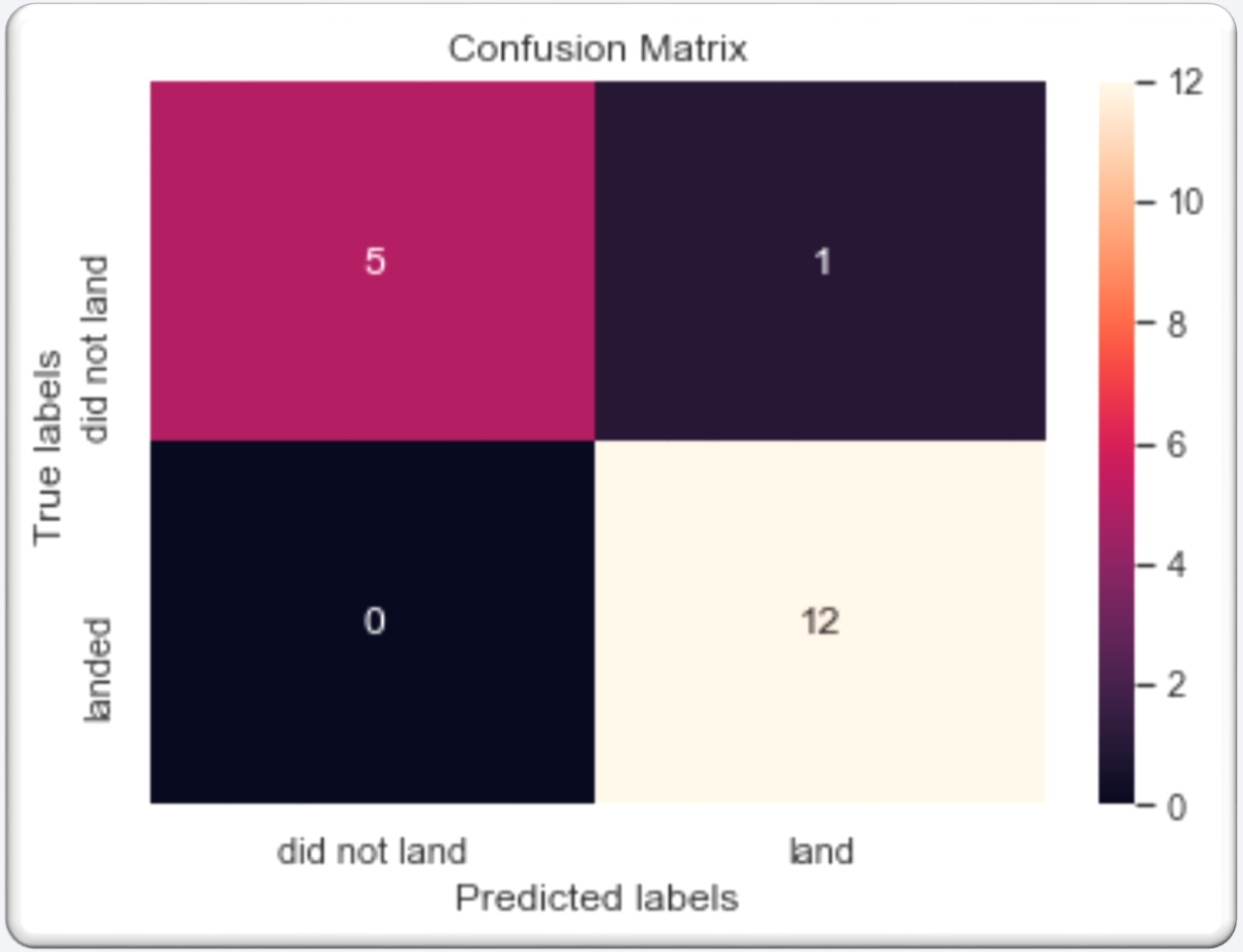
Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:

	Algorithm	best_scores	Score
0	Logistic Regression	0.846429	0.833333
1	Support Vector Machine	0.848214	0.833333
2	Decision Tree	0.885714	0.833333
3	KNearest Neighbor	0.848214	0.833333

The Decision Tree model has the highest classification accuracy

- The Accuracy Score is 94.44%
- The Best Score is 90.36%

Confusion Matrix



From the table presented previously, the best performing classification model is the Decision Tree model, with an accuracy of 88.57%.

This is explained by the confusion matrix, which shows only 1 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).

The other 17 results are correctly classified (5 did not land, 12 did land).

The Decision Tree classifier was able to classifier rockets that landed successfully correctly without errors

Conclusions

- As the number of flights increases, the rate of success at a launch site increases; though, most early flights was unsuccessful. We can say that, the success rate increased with experience.
  - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
  - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
  - After 2016, there was always a greater than 50% chance of a successful landing.

- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
    - The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
    - The 100% success rate in SSO is more impressive, with 5 successful flights.
    - The orbit types PO, ISS, and LEO, have more success with heavy payloads:
    - VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.

- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.
- The success for massive payloads (over 4000kg) is lower than that for low payloads.
- The best performing classification model is the Decision Tree model, with an accuracy of 88.57%.

Appendix

TASK 12

Find the method performs best:

```
[ ] # create list of algorithms
    algorithm_used = ['Logistic Regression', 'Support Vector Machine', 'Decision Tree', 'KNearest Neighbor']
    # create a list of best scores
    best_scores = [lr_best, svm_best, tree_best, knn_best]
    # creat a list of scores
    Score = [lr_score, svm_score, tree_score, knn_score]
    # create a list of column name
    column_names = ['Algorithm', 'best_scores', 'Score']
```

```
[ ] df = pd.DataFrame(list(zip(algorithm_used, best_scores, Score)), columns= column_names)
```

```
[ ] df.head()
```

TODO: For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster

```
#put launches into clusters and put onto map
marker_cluster = MarkerCluster()
site_map.add_child(marker_cluster) #note, this won't show up unless the individual launches are put onto the map


for index, row in spacex_df.iterrows():

    #define the position and launch site for each launch
    lat = row['Lat']
    long = row['Long']
    launch_site = row['Launch Site']

    #deine the launch marker for each launch
    launch_marker = folium.Marker(
        [lat, long],
        # Create an icon as a text label and colour depending on launch success/failure (marker colour)
        icon=folium.Icon(color='white', icon_color=row['marker_color'])
    )

    # Add the launch marker for each launch to the launch clusters (not the map, the launch clusters themselves)
    marker_cluster.add_child(launch_marker)

#show the map zoomed out with the centre at NASA
site_map
```

Thank you!